Chair for Algebraic and Logical
Foundations of Computer Science
TU Dresden

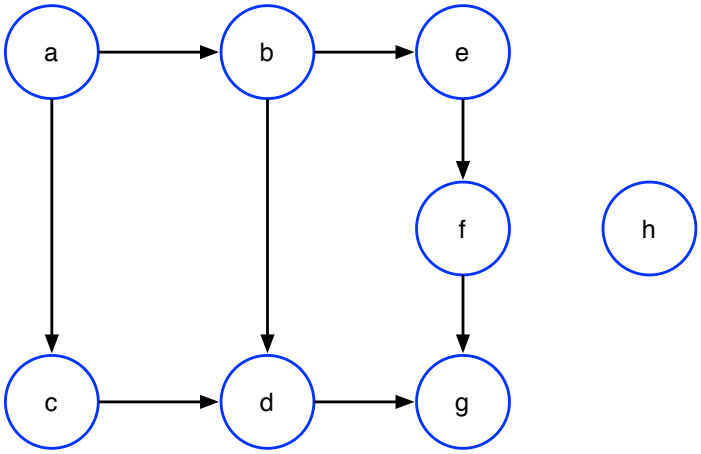
Winter Term 2021/2022

Logic Programming Engineering – Assignment 10

Dr.-Ing. Sascha Klüppelholz

Exercise 10.1 [Acyclic directed graphs]

Encode the following acyclic directed graph (acyclic digraph) by means of a Prolog program using predicates `nodes/1` that allows representing the set of all nodes and a predicate `arc/2` for the arcs of a directed graph.



a) Derive predicates `node/1` which evaluates to `true` if the given node is within the list of nodes (provided by `nodes/1`) and a predicate `arcs/1` that collects all arcs (given in terms of `arc/1`) in a set.

b) Provide a predicate `criteria/2` which is `true` if the out-degree of the given node is greater or equal to a given natural number. Use the SWI-Prolog predicate `partition/4` to partition the set of nodes into a set A where the out-degree is at least 2 and a set B collecting all nodes with out-degree lower than 2.

c) Consider the following four variants of a predicate intended for characterizing paths in acyclic digraphs

```
path1(X,Y) :- arc(X,Y).
path1(X,Y) :- arc(X,Z), path1(Z,Y).

path2(X,Y) :- arc(X,Y).
path2(X,Y) :- path2(X,Z), arc(Z,Y).

path3(X,Y) :- arc(X,Z), path3(Z,Y).
path3(X,Y) :- arc(X,Y).

path4(X,Y) :- path4(X,Z), arc(Z,Y).
path4(X,Y) :- arc(X,Y).
```

Predict and verify afterwards which of the above path predicates provide(s) the expected result, i.e., an enumeration of all node pairs (v,w) such that is a sequence of arcs from v to w. Use the following queries for testing.

```
?- path(c,f).
false

?- path(c,g).
true

?- path(X,e).
X = b; X = a.

?- path(e,Y).
Y = f; Y = g.

?- path(X,Y).
X = a,Y = b; X = a,Y = c; X = b,Y = d; X = b,Y = e; X = c,Y = d;
X = d,Y = g; X = e,Y = f; X = f,Y = g; X = a,Y = d; X = a,Y = e;
X = a,Y = g; X = a,Y = f; X = a,Y = g; X = a,Y = d; X = a,Y = g;
X = b,Y = g; X = b,Y = f; X = b,Y = g; X = c,Y = g; X = e,Y = g.
```

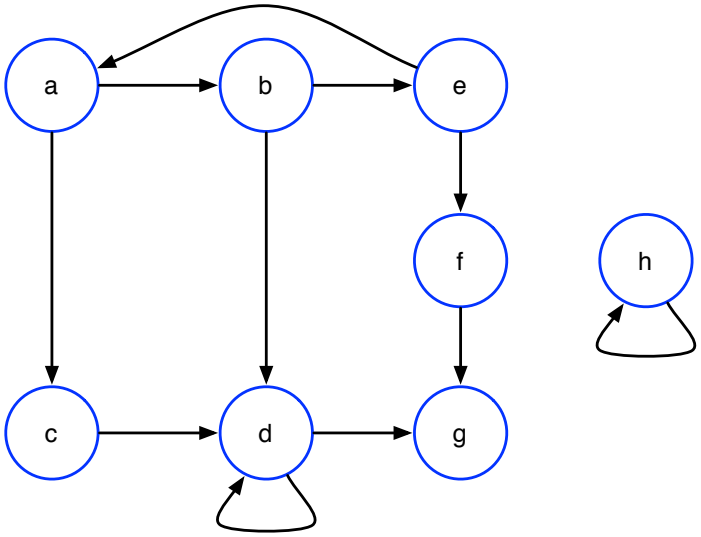
d) Provide a predicate `allPaths/1` that computes all paths from any starting state to any other state.

Example:

```
?- allPaths(P).
P = [a, b]; P = [a, c]; P = [b, d]; P = [b, e]; P = [c, d];
P = [d, g]; P = [e, f]; P = [f, g]; P = [a, b, d]; P = [a, b, e];
P = [a, b, d, g]; P = [a, b, e, f]; P = [a, b, e, f, g];
P = [a, c, d]; P = [a, c, d, g]; P = [b, d, g]; P = [b, e, f];
P = [b, e, f, g]; P = [c, d, g]; P = [e, f, g].
```

Exercise 10.2 [Directed graphs (digraphs)]

a) Does any of the predicates `path/2` and `allPaths/1` from Exercise 10.1 work for *cyclic digraphs* as well? Justify your answer. If the answer is no, fix the problem by refining the respective predicate such that works for cyclic digraphs as well. For testing use again the predicates `nodes/1` and `arc/2` from Exercise 10.1 and encode the following graph.



b) Write a predicate `reachable/2` returning a list of all reachable nodes for a starting node. Provide also predicates `dfs/2` and `bfs/2` that return lists of reachable nodes in a depth-first order and in a breadth-first order, respectively.

Examples:

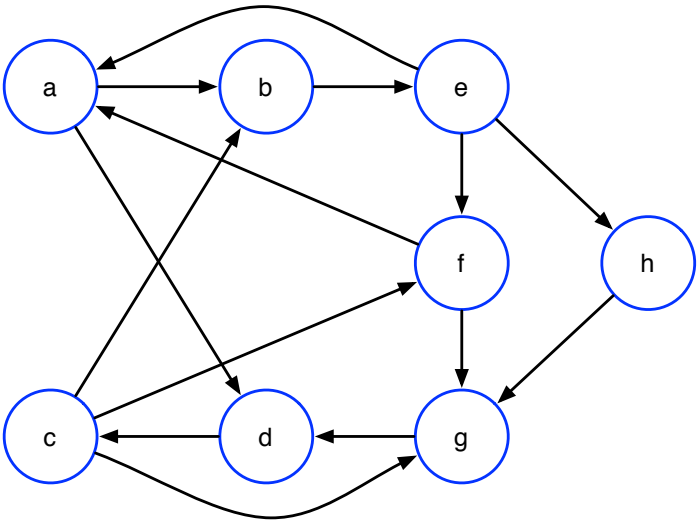
```
?- reachable(a,L).
L = [a, b, c, d, e, f, g].

?- reachable(h,L).
L = [h].

?- dfs(a,L).
L = [a, b, d, g, e, f, c].

?- bfs(a,L).
L = [a, b, c, d, e, g, f].
```

c) Write a predicate `maxPath/2` that enumerates for a given starting node all maximal paths, i.e., the paths that can not be further extended.

Example:

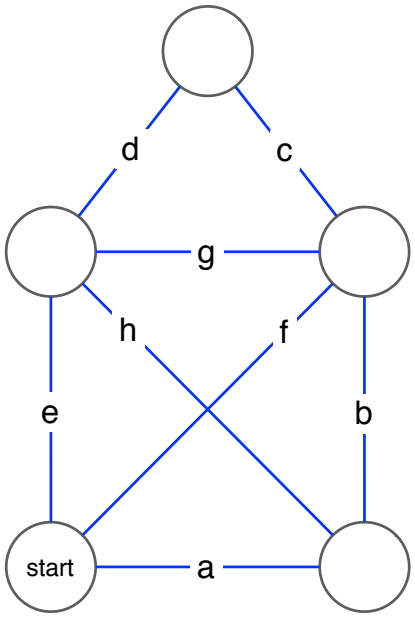
```
?- maxPath(a,L).
L = [a, b, d, g]; L = [a, b, e, f, g]; L = [a, c, d, g].
```

d) Provide predicates `hamiltonianPath/1` and `hamiltonianCycle/1` that enumerates the Hamiltonian paths and Hamiltonian cycles of a given digraph. What is the number of Hamiltonian paths and Hamiltonian cycles in the digraph depicted below?



Exercise 10.3 [Das Haus des Nikolaus]

"Das Haus des Nikolaus" is a puzzle for kids. The goal is to draw the following figure by starting at the node in the lower left, drawing lines to adjacent nodes until all eight edges a–h have been used exactly once without lifting the pen.



Provide a predicate `hausDesNikolaus/1` that enumerates all solutions. How many solutions are there?

Example:

```
?- hausDesNikolaus(P).
P = [e,d,c,g,h,a,f,b]; ...
```

Exercise 10.4 [Graph variant of the word guessing game]

Provide a variant of the Prolog program from Exercise 9.1 that works with finite directed graphs (finite digraphs) rather than with words. Your solution should introduce a new Prolog predicate `arc/2`. In particular your program should now provide the following features:

1. read/write database of nodes and arcs from/to a file
2. list all nodes and arcs currently in the database
3. add/delete a node or arc to/from the database
4. analyze the graph:
   a) write a predicate `degree/2` that determines the degree of a node, i.e., the number of arcs.
   b) provide a predicate `sortByDegree/1` that generates a list of all nodes of the graph, sorted in decreasing order according their degree.
5. end the program